VLSI DECOMPOSITIONS FOR DEBRUIJN GRAPHS

Sam Dolinar[1], Tsz-Mei Ko[2], and Robert McEliece[1,2*]

[1]Jet Propulsion Laboratory, and [2]Department of Electrical Engineering
California Institute of Technology
Pasadena, California 91125, USA

Abstract — A *C-chip VLSI decomposition of a graph* G is a collection of C isomorphic disjoint subgraphs of G (the "building blocks") which together contain all of G's vertices. The *efficiency* of such a decomposition is defined to be the fraction of edges of G that are in the building block. In this paper, motivated by the need to construct large Viterbi decoders, we study VLSI decompositions of deBruijn graphs. We obtain some strong necessary conditions for a graph to be a building block for a deBruijn graph, and some slightly more restrictive sufficient conditions which allow us to construct some efficient building blocks for deBruijn graphs.

1 The VLSI Decomposition Problem

Consider a graph G which represents an interconnection network for a complex circuit, such as a high-speed parallel computer, with the vertices of G representing arithmetic processors and the edges of G representing data paths connecting the processors. In modern VLSI technology, if the circuit is too large to fit on a single chip, it may be possible to build it by wiring together two or more appropriately designed chips. Each processor must then be placed on one of the chips, but the wires of the circuit may be either internal to the chips (intrachip wires) or external (interchip wires). For reasons of economy and simplicity, it is desirable, though not necessary, for the chips to be identical. Thus we are motivated to define a *VLSI decomposition* of a graph G as a collection of disjoint, isomorphic, subgraphs of G which together contain all of G's vertices, and a subset of its edges. We call the isomorphic subgraphs comprising the decomposition *building blocks* for the graph G, and we refer to the edges contained in the collection of subgraphs as *internal* edges. If a building block can be used to build any graph in a fixed set of graphs $\{G_n\}$, we call it a *universal building block* for $\{G_n\}$. It is normally desirable for the building block to include as many internal edges as possible, and so we define the *efficiency* of a VLSI decomposition of G as the fraction of the total number of edges in G which are internal edges.

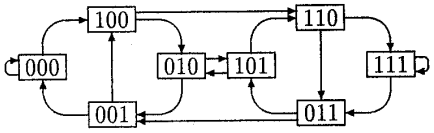

Figure 1: The deBruijn graph B_3

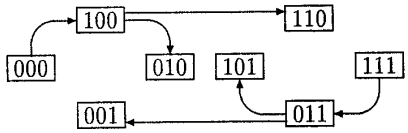

Figure 2: A two-chip VLSI decomposition of B_3

As an example, consider the deBruijn graph B_3 in Figure 1. (A formal definition of the deBruijn graph B_n will be given in Section 2). It contains 8 vertices and 16 edges. In Figure 2 we see a two-chip VLSI decomposition of B_3, in which the underlying building block contains four vertices and three edges. This decomposition has efficiency 3/8. It turns out that the building block in Figure 2 is a universal building block for the set $\{B_n\}_{n\geq 2}$. Indeed, this building block is just one of a large family of universal deBruijn building blocks we will describe in Theorem 3.

Although the VLSI decomposition problem is a very general one, in this paper we will focus entirely on the family of deBruijn graphs. We are interested in decomposing deBruijn graphs since they represent the circuit diagrams for fully parallel Viterbi decoders. Indeed, a constraint length K, rate $1/n$ convolutional code has the deBruijn graph B_{K-2} as the circuit diagram for its Viterbi decoder, and NASA is using a $K = 15$, rate $1/4$ convolutional code on the *Galileo* mission. As we will see, the methods described in this paper can be used to design a 64-chip VLSI decomposition of B_{13}

*The research described in this paper was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Tsz-Mei Ko's contribution was also supported by National Security Agency Grant No. MDA904-90-H-1007. Robert McEliece's contribution was also supported by AFOSR grant 91-0037 and a grant from Pacific Bell.

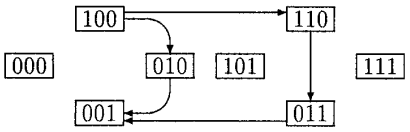

Figure 3: An unbalanced cycle of length 5 in B_3

with efficiency 0.754, which is being used by JPL design engineers to build a single-board Viterbi decoder for the *Galileo* code. (Earlier, but less efficient, VLSI decompositions for deBruijn graphs were presented in [1, 2]. Those led to a 256-chip VLSI decomposition of B_{13} of efficiency 0.563, which was used to design a multi-board decoder for the Galileo code.)

2 The DeBruijn Graph B_n

The binary deBruijn graph B_n can be defined as a directed graph with 2^n vertices, each represented with a binary n-tuple, and 2^{n+1} edges, each labelled with a binary $(n+1)$-tuple, where the edge $x_1x_2\cdots x_{n+1}$ is a directed edge from $x_2x_3\cdots x_{n+1}$ to $x_1x_2\cdots x_n$. In this section, we derive some properties of binary deBruijn graphs which are needed to study the VLSI decomposition for these graphs. (Other important properties can be found in [4, Secs. 2.2 and 6.2]).

Let $P = (P_1, P_2, \cdots, P_{l+1})$ be a sequence of $l+1$ vertices, and $E = (E_1, E_2, \cdots, E_l)$ be a sequence of l edges, in the graph B_n. If, for $i = 1, 2, \ldots, l$, E_i is an edge joining P_i and P_{i+1} (in either direction), we call (P, E) an *undirected path* of *length* l from P_1 to P_{l+1}. By tracing along the undirected path (P, E) in the order $P_1, P_2, \ldots, P_{l+1}$, its edge set E can be divided into two classes—forward edges (those in the same direction as the trace) and backward edges (those in the reverse direction). If there are f forward edges and b backward edges, we define the *net length* of the path to be $|f - b|$. A *directed path* is an undirected path with no backward edges. A *cycle* is an undirected path such that $P_1, P_2, \ldots, P_l$ are all distinct, but $P_{l+1} = P_1$. A cycle in which the number of forward edges equals the number of backward edges is called a *balanced* cycle. A cycle which is not balanced is called an *unbalanced* cycle. Figure 3 shows an unbalanced cycle of length 5 in B_3.

Theorem 1 *If X and Y are vertices in B_n, and l is a positive integer with $l \leq n$, then there cannot be more than one directed path of length l from X to Y.*

Proof: Let $X = x_1x_2\cdots x_n$. In a directed path from X to Y, the vertex immediately following X must be $y_1x_1x_2\cdots x_{n-1}$ for some $y_1 \in \{0, 1\}$. The next vertex must then be $y_2y_1x_1\cdots x_{n-2}$, and so on. Thus Y, which is the lth vertex in the path, must be of the form $Y = y_ly_{l-1}\cdots y_1x_1x_2\cdots x_{n-l}$. It follows that each vertex on the path, and hence the path itself, is uniquely determined by X and Y. ∎

Let us call an unbalanced cycle in B_n with r forward edges and s backward edges an (r, s) *cycle*. The following theorem shows that the number of vertex-disjoint (r, s) cycles in B_n is bounded above by a number independent of n.

Theorem 2 *In B_n, there are at most $2^l/l$ vertex-disjoint (r, s) cycles, where $l = r + s$ is the length of the cycle.*

Proof: If $l \geq n$, the theorem is trivially true, since B_n has 2^n vertices and thus at most $2^n/l \leq 2^l/l$ vertex-disjoint unbalanced cycles of length l. On the other hand, if $l < n$, let $A = a_1a_2\cdots a_n$ be a vertex which lies on an (r, s) cycle. Since a cycle may be traversed in either of two opposite directions, we may assume $r > s$ without loss of generality. After traversing the cycle and returning back to A, the substring $a_{s+1}a_{s+2}\cdots a_{n-r}$ will have been shifted to the right $r - s$ positions and so we have

$$A = x_1x_2\cdots x_ra_{s+1}a_{s+2}\cdots a_{n-r}y_1y_2\cdots y_s$$

Since this expression must be the same as the original label $a_1a_2\cdots a_n$, by equating the corresponding bits in positions $r+1$ through $n-s$, we obtain the equations

$$(1) \qquad a_{r+i} = a_{s+i} \quad (1 \leq i \leq n - s + r).$$

It follows from (1) that A is uniquely determined by the l values $x_1, \ldots, x_r$ and $y_1, \ldots, y_s$, so that there are at most $2^{r+s} = 2^l$ vertices that can lie on an (r, s) cycle. Since each (r, s) cycle contains exactly l vertices, it follows that a set of vertex-disjoint (r, s) cycles contains at most $2^l/l$ members. ∎

3 DeBruijn Building Blocks

From Theorems 1 and 2, we obtain two necessary conditions for deBruijn building blocks:

(i) A building block for B_n cannot contain two vertices X and Y such that there are two directed paths of the same length $l \leq n$ from X to Y.

(ii) A C-chip VLSI decomposition of B_n cannot have any unbalanced cycle of length $l < \log_2 C$ in its building block (since Theorem 2 implies that $C \leq 2^l/l \leq 2^l$).

We now restrict our attention to *universal deBruijn building blocks* (*UBBs*). A UBB is defined to be a building block that can be used to build any deBruijn graph B_n, with $n \geq k$. It immediately follows from (i) and (ii) that

(iii) A UBB cannot contain two vertices X and Y such that there are two directed paths of the same length from X to Y.

(iv) A UBB cannot contain any unbalanced cycle.

Surprisingly, any subgraph of B_k, with 2^k vertices, that satisfies (iii) and (iv), and the following weak additional requirement:

(v) Contains no undirected path of net length $> k$

is a universal deBruijn building block. (Note that condition (v) implies condition (iii) by Theorem 1.)

Theorem 3 *If U_k is a subgraph of B_k that has 2^k vertices and satisfies conditions (iv) and (v), then U_k is a UBB , i.e., it can be used to build any deBruijn graph B_n with $n \geq k$.*

We shall not give a proof of Theorem 3 in this paper. However, in the following section, we will describe an explicit algorithm for building B_n from 2^{n-k} copies of U_k. Of course, a proof of the correctness of this algorithm is equivalent to a proof of Theorem 3. (A formal proof of Theorem 3 can be found in [3].)

4 An Algorithm for Building B_n.

Let U_k be a subgraph of B_k that has 2^k vertices and satisfies conditions (iv) and (v). In this section we will describe an algorithm for building B_n with 2^{n-k} copies of U_k, which we call "chips". Each chip has an $n-k$ bit number, from $00\cdots0$ to $11\cdots1$. The algorithm runs as follows (in what follows, "$\oplus$" denotes modulo two addition):

(i) In U_k, choose an arbitrary vertex, say $x_1x_2\cdots x_k$. Label the corresponding vertex in chip number $a_1a_2\cdots a_{n-k}$ as $y_1y_2\cdots y_n$, where $y_i = a_i$ for $1 \leq i \leq n-k$ and $y_i = y_{i-(n-k)} \oplus x_{i-(n-k)}$ for $n-k < i \leq n$.

(ii) If, in some chip, there is a directed edge from an unlabelled vertex (corresponding to vertex $x_2x_3\cdots x_{k+1}$ in U_k) to a labelled vertex (corresponding to vertex $x_1x_2\cdots x_k$ in U_k) with label $y_1y_2\cdots y_n$, then label the unlabelled vertex as $y_2y_3\cdots y_{n+1}$, where $y_{n+1} = x_{k+1} \oplus y_{k+1}$.

(iii) If, in some chip, there is a directed edge from a labelled vertex (corresponding to vertex $x_2x_3\cdots x_{k+1}$ in U_k) with label $y_2y_3\cdots y_{n+1}$ to an unlabelled vertex (corresponding to vertex $x_1x_2\cdots x_k$ in U_k), then label the unlabelled vertex as $y_1y_2\cdots y_n$, where $y_1 = x_1 \oplus y_{n-k+1}$.

(iv) Repeat steps (ii) and (iii), in any order, until all the vertices in each chip have been labelled. (If U_k is not connected, it will be necessary to execute step (i) for each connected component of U_k.)

Example: We illustrate the above algorithm by using the UBB U_3 (shown in Figure 4) to generate four duplicate copies which together build B_5.

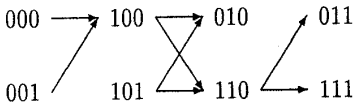

Figure 4: The most efficient U_3 building block

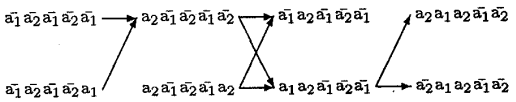

Figure 5: Four copies of U_k (with $a_1a_2 = 00, 01, 10, 11$) which together build B_5

Referring to Figure 4, we begin at step (i) by choosing vertex 110. The label of the corresponding vertex in chip number a_1a_2 must then be $a_1a_2\bar{a}_1\bar{a}_2\bar{a}_1$, since, by the rules of step (i),

$$\begin{aligned} y_1 &= a_1, \quad y_2 = a_2, \\ y_3 &= y_1 \oplus x_1 = a_1 \oplus 1 = \bar{a}_1, \\ y_4 &= y_2 \oplus x_2 = a_2 \oplus 1 = \bar{a}_2, \\ y_5 &= y_3 \oplus x_3 = \bar{a}_1 \oplus 0 = \bar{a}_1. \end{aligned}$$

We can now execute step (ii), since in chip number a_1a_2 there is a directed edge from an unlabelled vertex (corresponding to 101) to a labelled vertex (corresponding to 110). By the rules of step (ii) the vertex in chip a_1a_2 corresponding to 101 must be labelled $a_2\bar{a}_1\bar{a}_2\bar{a}_1a_2$ since $y_{n+1} = y_6 = \bar{a}_2 \oplus 1 = a_2$. Similarly, the vertex corresponding to 100 in chip a_1a_2 must be labelled $a_2\bar{a}_1\bar{a}_2\bar{a}_1\bar{a}_2$.

There are now two possible ways to apply step (iii) to label the vertex corresponding to 010, since 010 is connected to both 100 and 101, both of which have already been labelled. However, both ways produce the same label, viz, $\bar{a}_1a_2\bar{a}_1\bar{a}_2\bar{a}_1$. (In general, it can be shown that consistency in the labelling algorithm always holds for a graph satisfying the conditions of Theorem 3.)

Continuing in this way, we obtain the complete labelling of the four chips shown in Figure 5. (The four copies are obtained by letting $a_1a_2 =$ 00, 01, 10, and 11.)

5 The Most Efficient Known UBBs

Theorem 3 says that any subgraph of B_k with no unbalanced cycle and no undirected path of net length $> k$ is a universal deBruijn building block U_k. There are many such UBBs, but we are only interested in the "efficient" ones. Now the efficiency e_k of a UBB U_k is independent of the size of the deBruijn graph which it is used to build. Indeed, if there are E_k edges in U_k and 2^{n-k} copies of U_k are used to build B_n, then there

k	E_k	e_k	c_k
1	1	0.250	1.500
2	3	0.375	1.875
3	8	0.500	2.000
4	19	0.594	2.031
5	43	0.672	1.969
6	92	0.719	1.969
7	193	0.754	1.969
8	398	0.777	2.004

Table 1: The most efficient known U_k building blocks, for $1 \leq k \leq 8$. In this table E_k denotes the number of edges, e_k and c_k are defined in equations 2 and 3.

are a total of $2^{n-k}E_k$ internal edges out of 2^{n+1} total edges in B_n. Thus the efficiency is

$$e_k = \frac{2^{n-k}E_k}{2^{n+1}} = \frac{E_k}{2^{k+1}} \tag{2}$$

independent of n. Table 1 lists the number of edges in the most efficient U_k we have been able to find, using ad hoc methods, for $1 \leq k \leq 8$.[1] We have been able to show by exhaustive search that the entries for $1 \leq k \leq 4$ are optimal, but for larger values of k, improvements may be possible. For $k = 1$, the optimal building block consists of a single edge. The optimal U_2 and U_3 building blocks are shown in Figures 2 and 4. Figure 6 shows the best known U_7 building block that is being used to build the single board Viterbi decoder for the Galileo code.

If e_k^* denotes the maximum possible efficiency of a U_k building block, it is natural to study the asymptotic behavior of e_k^* as $k \to \infty$. Towards this end, we define the quantity c_k^* by the formula

$$e_k^* = 1 - \frac{c_k^*}{k+1}. \tag{3}$$

By property (iv) in Section 3, no directed cycle can appear in a VLSI decomposition of B_k by universal building blocks. A theorem of Mykkeltveit [5] shows that the minimum number of edges that must be removed from B_k to break all directed cycles is $\sim 2^k/k$. This result implies $\liminf_{n\to\infty} c_k^* \geq 1$. On the other hand, Schwabe [6] has proved, using the methods developed in [1], that $\limsup_{n\to\infty} c_k^* \leq 8$. However, if we look at Table 1, where c_k for the most efficient known U_k building blocks is tabulated for $1 \leq k \leq 8$, something much stronger seems to be true, and we conjecture that $\lim_{k\to\infty} c_k^* = 2$.

[1] The building blocks for $k = 5$ and $k = 8$ were first discovered by a clever computer search algorithm developed by Gordon Oliver.

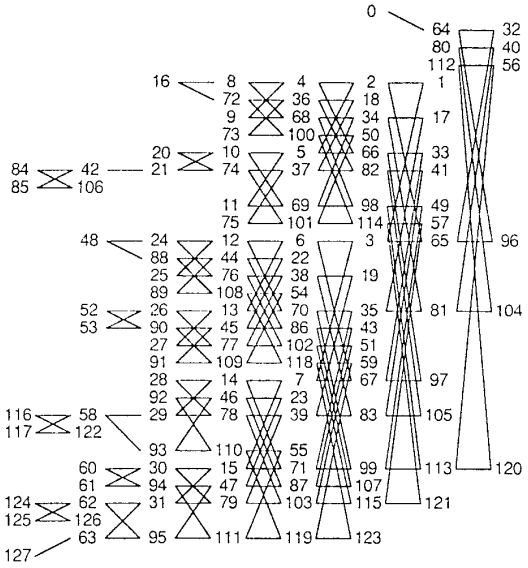

Figure 6: The most efficient known U_7 universal building block, with 193 edges. In this figure, the binary strings are represented by their decimal equivalents.

References

[1] O. Collins, S. Dolinar, R. McEliece, and F. Pollara, "A VLSI Decomposition of the DeBruijn Graph," *J. Assoc. Comp. Mach., in press.*

[2] O. Collins, F. Pollara, S. Dolinar, and J. Statman, "Wiring Viterbi Decoders (Splitting DeBruijn Graphs)," *JPL TDA Progress Report* 42-96 (1988), pp. 93–103.

[3] S. Dolinar, T.-M. Ko, and R. McEliece "Some VLSI Decompositions of the DeBruijn Graph," *Symp. Discrete Math. Appl.*, Netherlands, Sept, 1992.

[4] S. Golomb, *Shift Register Sequences (Rev. Ed.).* Laguna Hills, Calif.: Aegean Park Press, 1982.

[5] J. Mykkeltveit, "A proof of Golomb's conjecture for the deBruijn graph," *J. Comb. Theory (B)* 13 (1972), pp. 40–45.

[6] E. Schwabe, private communication, February 15, 1991.